# Apophysis Back to Basics: Rendering Optimisation #1

## Introduction

The trio of parameters quality (Q), oversample (OS) and filter radius (FR) are the ones that essentially determine the *physical quality* of the render. This tutorial takes a definitive look at how these parameters work together.

## Definitions

These are not formal definitions, rather they give a rough functional summary of each parameter.

**Quality** (Q) – effectively, the iterations for a given 'canvas' size, hence it determines the 'hit density' of points.

**Oversample** (OS) – traditionally, this is the reduction ratio (linear dimension) for the picture rendered in memory to the final stored file. Hence, OS=2 would mean that for a final picture of 1024px square, the computer actually renders one at 2048px square. Anti-aliasing is effectively carried out on pixels.

In Apophysis, the process is slightly different. Instead of simply rendering larger, the image *is* rendered in larger *but* at reduced *effective* quality. For instance, rendering at OS=2, the linear dimension is doubled whilst same number of hits (quality value) is spread over the increased area, thus the *effective* quality (hit density) is reduced as the square of the oversample. However, this intermediate stage is never observed. This method only actually achieves anti-aliasing in conjunction with the next parameter, and it is here that its superiority to basic anti-aliasing becomes apparent.

**Filter radius** (FR) – the size of the blur filter applied during the final stage of rendering.

## The motivation

Sometimes rendering is a compromise: you want to squeeze out maximum detail, yet you want to keep the curves smooth and avoid the dreaded 'jaggies', the pixel-stepping that dogs many an otherwise decent render. I never used to bother with oversample, preferring to render large and scale down using my own patented (;P) multi-step method of reduction to force better anti-aliasing. Then a comment by Joel Faber on the Apo mailing list switched on a light for me: oversample works *in conjunction with* filter radius!

It's worth stating now that requirements for prints are different from those for a completed electronic display image: prints can generally manage without oversample, especially at higher (300dpi) quality, due to the size reduction inherent in the printing process.

The chosen setting for oversample will depend mainly on available RAM (required RAM increases in proportion to the square of the value, so: 2 = 4x; 3 = 9x; 4 = 16x). Essentially, the required RAM should not exceed the available RAM, otherwise the fractal will be rendered in strips, the number required being the multiplier of the time taken. Other factors:

- image size (number of pixels directly proportionate to the RAM requirement)
- multitasking requirements (there should be sufficient spare RAM to perform other tasks if required)

The following table demonstrates the relationships with actual figures:

| Width (px) | Height (px) | Oversample | RAM (MB) |
|---:|---:|---:|---:|
| 640 | 480 | 1 | 4 |
| 640 | 480 | 2 | 18 |
| 640 | 480 | 3 | 42 |
| 640 | 480 | 4 | 75 |
| 1280 | 960 | 4 | 300 |

Hence, 1GB RAM should suffice to render a decently sized fractal with up to 4x oversample and still leave some RAM for other tasks. What follows is based on personal experience with 4x oversample. It should be noted that oversample, provided that the fractal is rendered in a single strip, does not unduly affect the time taken (it will increase render time very slightly for the extra final processing). The parameter that affects render time is quality, the relationship being one of direct proportionality. Say Q=500 takes 4 minutes, so Q=2000 will take 16 minutes.

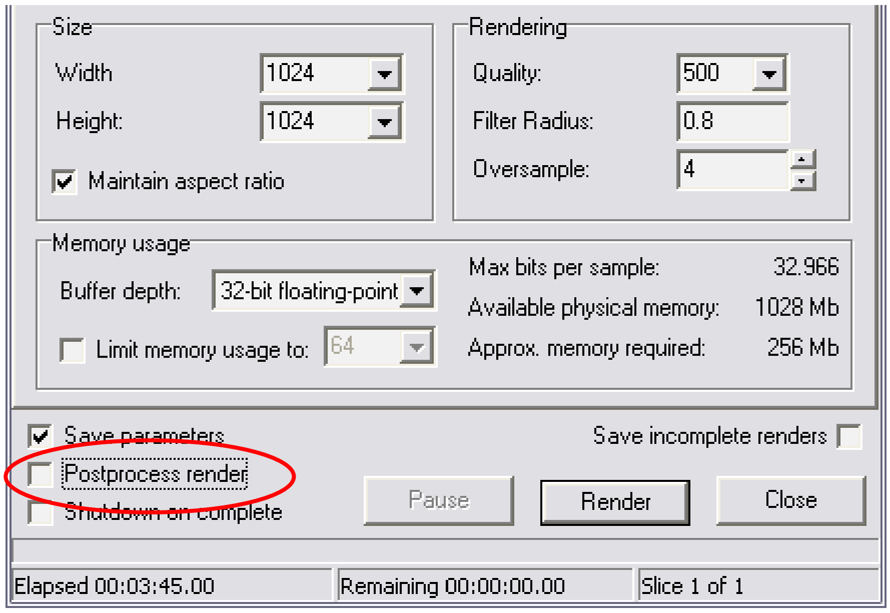
A quick summary on factors affecting render time and memory requirements:

**Render time = k1 x width x height x quality**

**Memory requirement = k2 x width x height x oversample$^2$**

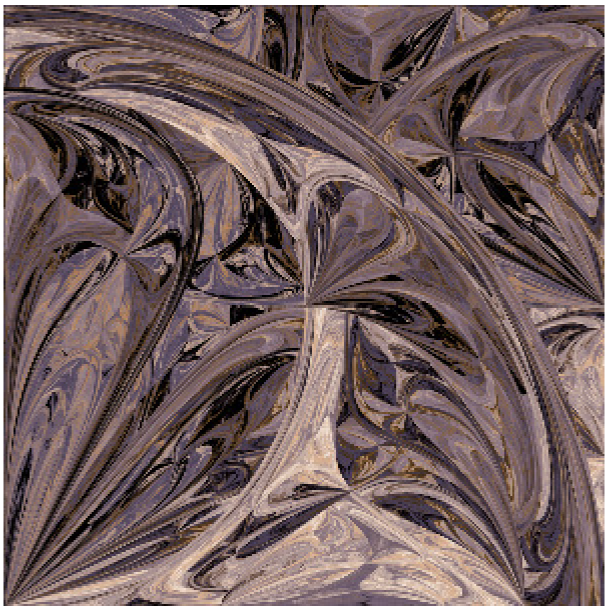
Where k1, k2 are system constants.

Now we've established a default value for oversample, it's time to examine the effect of filter radius. I usually perform a test render at Q=500 and FR=0.4. Sometimes Q=500 suffices for the final, sometimes I go as high as 10000. You may choose to check the 'Postprocess render' box in the 'Render to Disk' dialogue, thus enabling direct preview of the effect of changing the FR before completing the render.


Size
Width 1024
Height: 1024
[✓] Maintain aspect ratio
Rendering
Quality: 500
Filter Radius: 0.8
Oversample: 4
Memory usage
Buffer depth: 32-bit floating-point
[ ] Limit memory usage to: 64
Max bits per sample: 32.966
Available physical memory: 1028 Mb
Approx. memory required: 256 Mb
[✓] Save parameters
[ ] Postprocess render
[ ] Shutdown on complete
Save incomplete renders [ ]

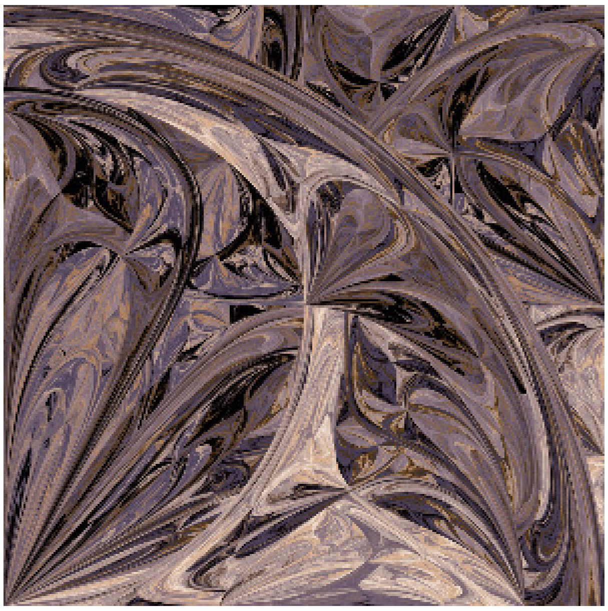
Elapsed 00:03:45.00
Remaining 00:00:00.00
Slice 1 of 1

Here's an example of changing FR values with a particularly jagged style:



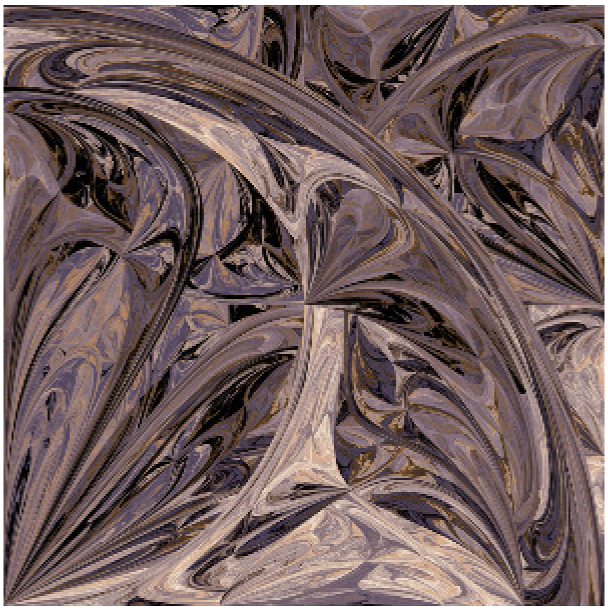
FR = 0.1

Very jagged.



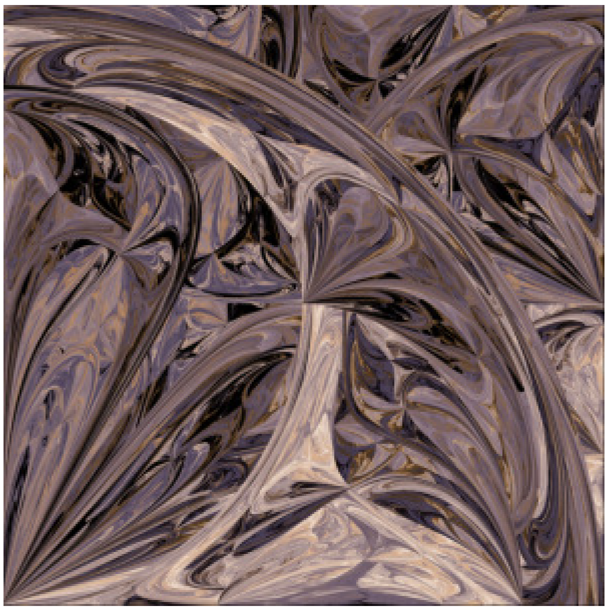
FR = 0.2

Still very jagged.



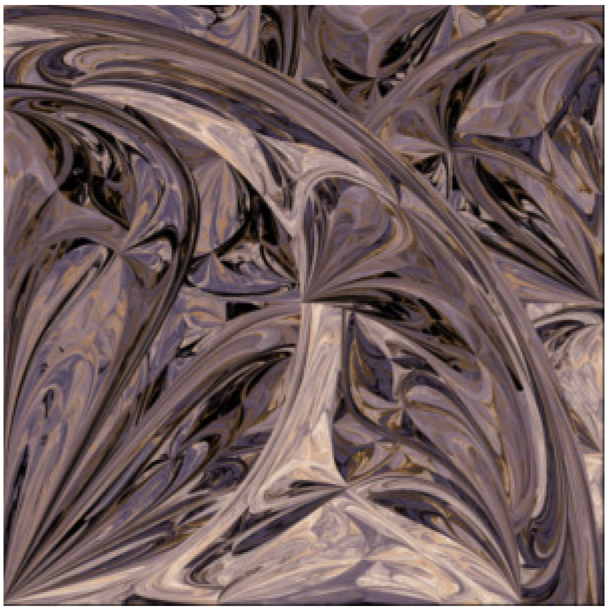
FR = 0.4

Still jagged, but beginning to show improvement.



FR = 0.8

Almost there…



FR = 1.0

That’ll do!

Of course, you can increase the value above 1.0, but I can't see the likelihood of requiring above 1.2 for OS=4. The aim of the exercise is to find the minimum filter radius commensurate with elimination of stepping. If you don’t use the 'Postprocess render' option, **always** perform a test render (or as many as necessary), at a reduced size if speed is an issue, to establish this value. Experience will enable correct choice on the first or second attempt.

So, you've eliminated those jagged steps but the detail's just not there any more? Sharpen the image using your image manipulation software. Try values around the 5 - 10% level to begin. It's interesting to analyse the process here: it seems as though we're simply undoing the blur we put in, but it's not so. Because the blur from the FR is applied in conjunction with the OS, then its effect is qualitatively different from a sharpen filter applied to the finished fractal.

http://ideviant.deviantart.com/

Special thanks to banana-tree (http://banana-tree.deviantart.com/) for finally clearing up for me how oversample works.